# Configurable and Scalable High Throughput Turbo Decoder Architecture for Multiple 4G Wireless Standards*

Yang Sun †, Yuming Zhu ‡, Manish Goel ‡, and Joseph R. Cavallaro †
†ECE Department, Rice University. 6100 Main, Houston, TX 77005
‡DSPS R&D Center, Texas Instruments. 12500 TI Blvd MS 8649, Dallas, TX 75243
Email: *ysun@rice.edu, y-zhu@ti.com, goel@ti.com, cavallar@rice.edu*

## Abstract

*In this paper, we propose a novel multi-code turbo decoder architecture for 4G wireless systems. To support various 4G standards, a configurable multi-mode MAP (maximum a posteriori) decoder is designed for both binary and duo-binary turbo codes with small resource overhead (less than 10%) compared to the single-mode architecture. To achieve high data rates in 4G, we present a parallel turbo decoder architecture with scalable parallelism tailored to the given throughput requirements. High-level parallelism is achieved by employing contention-free interleavers. Multi-banked memory structure and routing network among memories and MAP decoders are designed to operate at full speed with parallel interleavers. We designed a very low-complexity recursive on-line address generator supporting multiple interleaving patterns, which avoids the interleaver address memory. Design trade-offs in terms of area and power efficiency are explored to find the optimal architectures. A 711 Mbps data rate is feasible with 32 Radix-4 MAP decoders running at 200 MHz clock rate.*

## 1. Introduction

The approaching fourth-generation (4G) wireless systems are projected to provide 100 Mbps to 1 Gbps speeds by 2010, which consequently leads to orders of magnitude increases in the expenditure of the computing resources. The high throughput turbo codes [1] are required in many 4G communication applications. 3GPP Long Term Evolution (LTE) and IEEE 802.16e WiMax are two examples. As some 4G systems use different types of turbo coding schemes (e.g. binary codes in 3GPP LTE and duo-binary codes in WiMax), a general solution to supporting multiple code types is to use programmable processors. For example, a 2 Mbps turbo decoder implemented on a DSP processor is proposed in [2]. Also, authors in [3] and [4] develop a multi-code turbo decoder based on SIMD processors, where a 5.48 Mbps data rate is achieved in [3] and a 100 Mbps data rate is achieved in [4] at a cost of 16 processors. While these programmable SIMD/VLIW processors offer great flexibilities, they have several disadvantages notably higher power consumption and less throughput than ASIC solutions. A turbo decoder is typically one of the most computation-intensive parts in a 4G receiver, therefore it is essential to design an area and power efficient turbo decoder in ASIC. However, to the best of our knowledge, supporting multi-code (both simple binary and double binary codes) in ASIC is still lacking in the literature.

In this work, we propose an efficient VLSI architecture for turbo decoding. This architecture can be configured to support both simple and double binary turbo codes up to eight states. We address the memory collision problems by applying new highly-parallel interleavers. The MAP decoder, memory structure and routing network are designed to operate at full speed with the parallel interleaver. The proposed architecture meets the challenge of multi-standard turbo decoding at very high data rates.

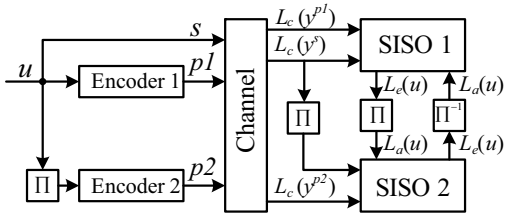



**Figure 1. Turbo encoder/decoder structure**

## 2. MAP decoding algorithm

The turbo decoding concept is functionally illustrated in Fig. 1. The decoding algorithm is called the maximum *a*

*This work was supported by the Summer Student Program 2007, Texas Instrument Incorporated.

*posteriori* (MAP) algorithm and is usually calculated in the log domain [5]. During the decoding process, each SISO decoder receives the intrinsic log-likelihood ratios (LLRs) from the channel and the extrinsic LLRs from the other constituent SISO decoder through interleaving ($\Pi$) or deinterleaving ($\Pi^{-1}$). Consider a decoding process of simple binary turbo codes, let $s_k$ be the trellis state at time $k$, then the MAP decoder computes the LLR of the *a posteriori* probability (APP) of each information bit $u_k$ by

$$\Lambda(\hat{u}_k) = \max_{\mathbf{u}:u_k=1}^{*}\{\alpha_{k-1}(s_{k-1}) + \gamma_k(s_{k-1}, s_k) + \beta_k(s_k))\} - \max_{\mathbf{u}:u_k=0}^{*}\{\alpha_{k-1}(s_{k-1}) + \gamma_k(s_{k-1}, s_k) + \beta_k(s_k))\}$$

where $\alpha_k$ and $\beta_k$ represent forward and backward state metrics, respectively, and are computed recursively as:

$$\alpha_k(s_k) = \max_{s_{k-1}}^{*}\{\alpha_{k-1}(s_{k-1}) + \gamma_k(s_{k-1}, s_k)\} \quad (1)$$

$$\beta_k(s_k) = \max_{s_{k+1}}^{*}\{\beta_{k+1}(s_{k+1}) + \gamma_k(s_k, s_{k+1})\} \quad (2)$$

The second term $\gamma_k$ is the branch transition probability and is usually referred to as a branch metric (BM). The $\max^*$ function, defined as $\max_e^*(f(e)) = \ln(\sum_e \exp(f(e)))$, is typically implemented as a $\max$ followed by a correction term [5]. To extract the extrinsic information, $\Lambda(\hat{u}_k)$ is split into three terms: extrinsic $L_e(\hat{u})$, a priori $L_a(u_k)$ and systematic $L_c(y_k^s)$ as: $\Lambda(\hat{u}_k) = L_e(\hat{u}_k) + L_a(u_k) + L_c(y_k^s)$.

## 2.1. Unified Radix-4 decoding algorithm

For binary turbo codes, the trellis cycles can be reduced 50% by applying the one-level look-ahead recursion [6][7] as illustrated in Fig. 2. Radix-4 $\alpha$ recursion is then given by:

$$\alpha_k(s_k) = \max_{s_{k-1}}^{*}\left\{\max_{s_{k-2}}^{*}\{\alpha_{k-2}(s_{k-2}) + \gamma_{k-1}(s_{k-2}, s_{k-1})\} + \gamma_k(s_{k-1}, s_k)\right\} = \max_{s_{k-2}, s_{k-1}}^{*}\{\alpha_{k-2}(s_{k-2}) + \gamma_k(s_{k-2}, s_k)\} \quad (3)$$

where $\gamma_k(s_{k-2}, s_k)$ is the new branch metric for the two-bit symbol $\{u_{k-1}, u_k\}$ connecting state $s_{k-2}$ and $s_k$:

$$\gamma_k(s_{k-2}, s_k) = \gamma_{k-1}(s_{k-2}, s_{k-1}) + \gamma_k(s_{k-1}, s_k) \quad (4)$$

Similarly, Radix-4 $\beta$ recursion is computed as:

$$\beta_k(s_k) = \max_{s_{k+2}, s_{k+1}}^{*}\{\beta_{k+2}(s_{k+2}) + \gamma_k(s_k, s_{k+2})\} \quad (5)$$

Since the Radix-4 algorithm is based on the symbol level, we must define a symbol probability (or reliability):

$$L(\psi_{ij}) = \max_{s_{k-2}, s_k}^{*}\{\alpha_{k-2}(s_{k-2}) + \gamma_k^{ij} + \beta_k(s_k)\} \quad (6)$$

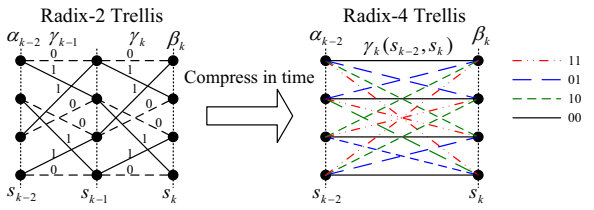



**Figure 2. 4-state trellis merge example**

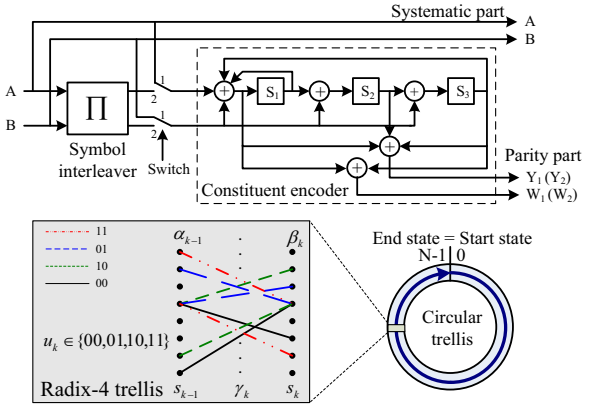



**Figure 3. Duo-binary convolutional encoder**

where $\gamma_k^{ij}$ is the branch transition probability with $u_{k-1} = i$ and $u_k = j$, for $i, j \in (0, 1)$, then the bit LLRs for $u_{k-1}$ and $u_k$ are computed as:

$$\Lambda(\hat{u}_{k-1}) = \max^{*}(L(\psi_{10}), L(\psi_{11})) - \max^{*}(L(\psi_{00}), L(\psi_{01}))$$

$$\Lambda(\hat{u}_k) = \max^{*}(L(\psi_{01}), L(\psi_{11})) - \max^{*}(L(\psi_{00}), L(\psi_{10}))$$

For duo-binary codes, they differ from binary codes mainly in the trellis terminating scheme and the symbol-wise decoding [8]. The duo-binary trellis is closed as a circle with the start state equal to the end state, this is also referred to as a tail-biting scheme, which is shown in Fig. 3. The decoding algorithm for duo-binary is inherently based on the Radix-4 algorithm [8], hence the same Radix-4 $\alpha$, $\beta$ and $L(\varphi)$ function units (3)(5)(6) can be applied to the duo-binary codes in a straightforward manner. The unique parts are the branch metrics $\gamma^{ij}$ calculations and tail-biting handling. Moreover, three LLRs must be calculated for duo-binary codes: $\Lambda^1(\hat{u}_k) = L_k(\psi_{01}) - L_k(\psi_{00})$, $\Lambda^2(\hat{u}_k) = L_k(\psi_{10}) - L_k(\psi_{00})$ and $\Lambda^3(\hat{u}_k) = L_k(\psi_{11}) - L_k(\psi_{00})$.

## 3. Unified Radix-4 MAP decoder architecture

Based on the justification in section 2.1 that both binary and duo-binary codes can be decoded in an unified way, we propose a configurable Radix-4 Log-MAP decoder architecture to perform both types of decoding operations. To

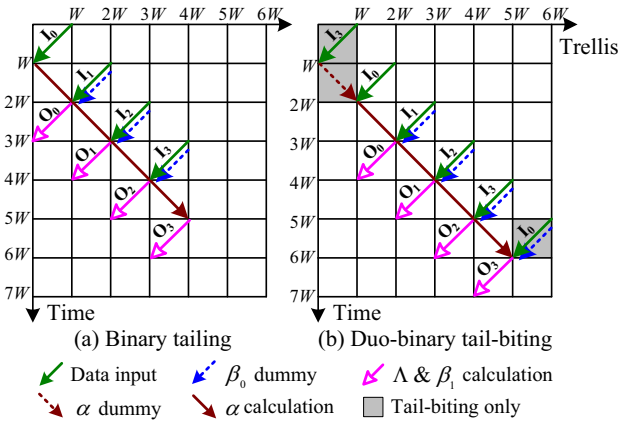



**Figure 4. Sliding window tile chart**

efficiently implement the Log-MAP algorithm, an overlapping sliding window technique [9] is employed.

We generalize the two types of decoding operations into one unified flow which is shown in Fig. 4. This decoding operation is based on three recursion units, two used for the backward recursions (dummy $\beta_0$ and effective $\beta_1$), and one for forward recursion ($\alpha$). Each recursion unit contains full-parallel ACSA (Add-Compare-Select-Add) operators. To reduce decoding latency, data in a sliding window is fed into the decoder in the reverse order; $\alpha$ unit is working in the natural order; dummy $\beta_0$, effective $\beta_1$ and $\Lambda$ units are working in the reverse order as shown in Fig. 4. This leads to a decoding latency of $2W$ for binary codes and $3W$ for duo-binary codes, where $W$ is the sliding window length. Duo-binary codes have longer latency because the start trellis states are not initialized to 0 but equal to the end states, hence an additional acquisition is needed to obtain the initial states' $\alpha$ metrics (see Fig. 4(b)). Similarly, an additional acquisition is also required for the end states' $\beta$ metrics, which however will not cause any additional delays.

Fig. 5 shows the proposed Radix-4 Log-MAP decoder architecture. Three scratch RAMs (each has a depth of $W$) were used to store the input LLRs (systematic, parity and *a priori*). Three branch metric calculation (BMC) units are devoted to compute the branch metrics for the $\alpha$, $\beta_0$ and $\beta_1$ function units. To support multi-code, the decoder employs configurable BMCs and configurable $\alpha$ and $\beta$ function units which can support multiple transfer functions by configuring the routing (multiplexors) blocks (top-right Fig. 5). Each $\alpha$ and $\beta$ unit consists of 8 ACSA units so they can support up to 8 state turbo decoding. The Radix-4 ACSA unit is implemented with $\max^*$ trees. Since we want to support both binary and duo-binary decoding, the extrinsic $\Lambda$ function unit as depicted in the bottom part of Fig. 5 contains both bit LLR and symbol LLR generation logic. After a latency of $2W$ or $3W$, the $\Lambda$ unit begins to generate soft

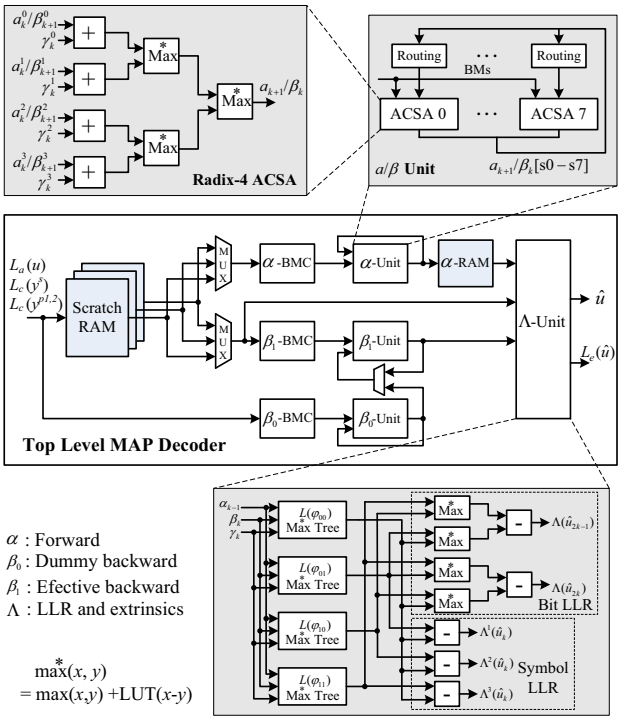



**Figure 5. Unified MAP decoder architecture**

LLRs $\Lambda(\hat{u})$ and extrinsic information $L_e(\hat{u}_k)$ in real time.

In this architecture, many parts of the logic circuits can be shared. For example, the $\alpha$, $\beta$ and $L(\varphi)$ units, the scratch and $\alpha$ RAMs can be easily shared between two operations. Table 1 compares the resource usage for a multi-code architecture and a single-code architecture, in which $M$ is the number of states in the trellis, $B_m$, $B_b$, $B_c$ and $B_e$ are the precisions of state metrics, branch metrics, channel LLRs and extrinsic LLRs, respectively. From table 1, we see the overhead for adding flexibility is very small (about 7%).

**Table 1. Hardware complexity comparison**

| | Multi-code | Single-code |
|---|---|---|
| Storage (bits) | $(9B_e + 12B_c + MB_m)W$ | $(9B_e + 12B_c + MB_m)W$ |
| $B_m$-bit $\max^*$ † | $(25/2)M + 4$ | $(25/2)M$ |
| 1-bit adder | $16MB_m + 10MB_b$ | $16MB_m + 10MB_b$ |
| 1-bit flip-flop | $5MB_m + 2MB_b$ | $5MB_m + 2MB_b$ |
| 1-bit mux | $16MB_m + 16MB_b$ | $3MB_m$ |
| Normalized area | 1.0 | 0.93 |

† 1 four-input $\max 4^*$ is counted as 3 two-input $\max^*$
1 eight-input $\max 8^*$ is counted as 7 two-input $\max^*$

The fixed-point word length chosen for this decoder is $B_c$=6, $B_e$=7 and $B_{m/b}$=10. The sliding window length $W$ is programmable and can be up to 64. Table 2 summarizes the synthesize results on a 65nm technology at 200MHz clock. Compared to the Radix-4 MAP decoder in [6] which

contains 410K gates of logic, our architecture achieves better flexibility by supporting multiple code types at low hardware overhead.

**Table 2. Area distribution of the MAP decoder**

| Blocks | Gate count |
|---|---|
| $\alpha$-processor (including $\alpha$-BMU) | 30.8K gates |
| $\beta$-processor x2 (including $\beta$-BMUs) | 66.2K gates |
| $\Lambda$-processor | 37.3K gates |
| $\alpha$-RAM | 2.560K bits |
| Scratch RAMs x3 | 4.224K bits |
| Control logic | 13.4K gates |

## 4. Low complexity interleaver architecture

One of the main obstacles to parallel turbo decoding is the interleaver parallelism due to the memory access collision problems. For example, in [10] memory collisions are solved by having additional write buffers. However, recently, new contention-free interleavers have been adopted by 4G standards, such as QPP interleaver [11] in 3GPP LTE and ARP interleaver [12] in WiMax. The simplest approach to implement an interleaver is to store all the interleaving patterns in ROMs. However, this approach becomes impractical for a turbo decoder supporting multiple block sizes and multiple standards. In this section, we propose an unified on-line address generator with two recursion units supporting both QPP and ARP interleavers.

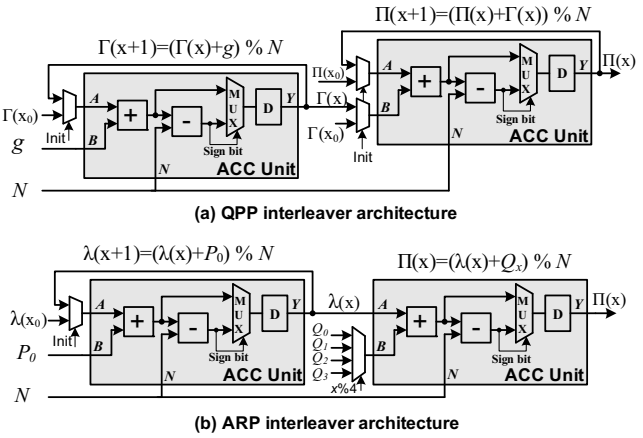



**Figure 6. Low-complexity unified interleaver**

We first look at the QPP interleaver. Given an information block length $N$, the $x$-th interleaved output position is given by $\Pi(x) = (f_2x^2 + f_1x) \bmod N, 0 \le x, f_1, f_2 < N$. This can be computed recursively as:

$$\begin{aligned}\Pi(x+1) &= \big((f_2x^2 + f_1x) + (2f_2x + f_1 + f_2)\big) \bmod N \\ &= (\Pi(x) + \Gamma(x)) \bmod N \end{aligned} \quad (7)$$

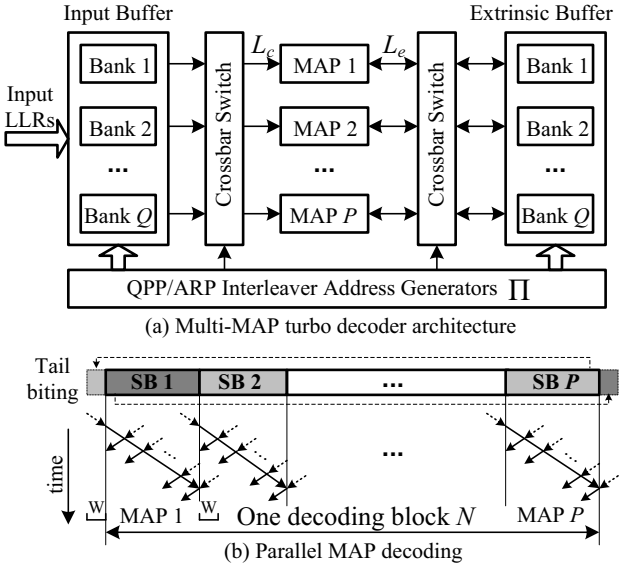



**Figure 7. Parallel decoder architecture**

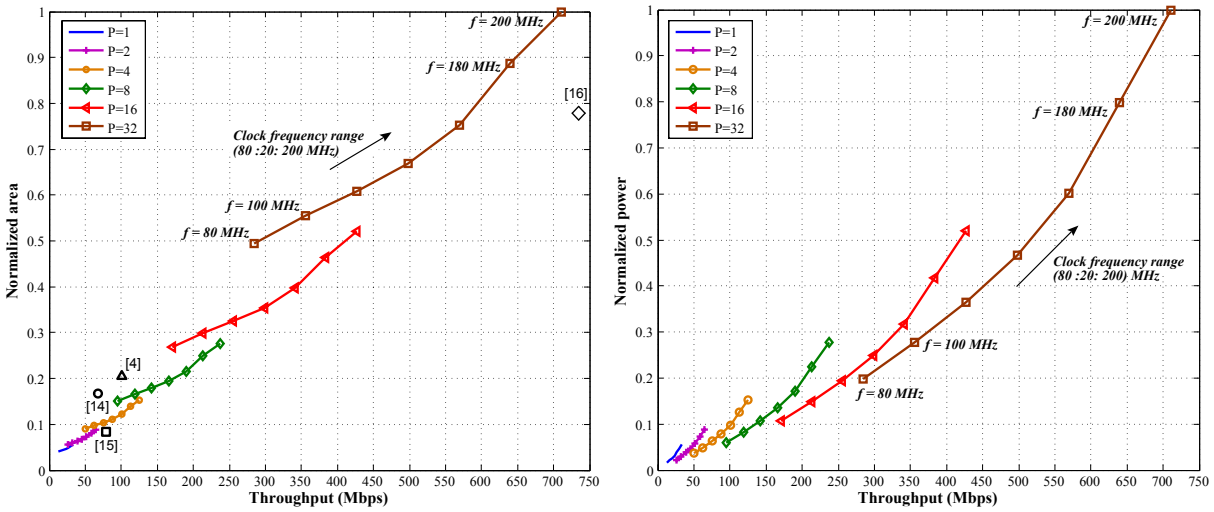
where $\Gamma(x) = (2f_2x + f_1 + f_2) \bmod N$, and it can also be computed recursively as: $\Gamma(x+1) = (\Gamma(x) + g) \mod N$, where $g = 2f_2$. Since $\Pi(x)$, $\Gamma(x)$ and $g$ are all smaller than $N$, the QPP interleaver can be efficiently implemented by cascading two Add-Compare-Choose (ACC) units as shown in Fig. 6(a), which avoids the expensive multipliers and dividers.

For the ARP interleaver, it employs a two-step interleaving. In the first step it switches the alternate couples as $[B_x, A_x]$=$[A_x, B_x]$ if $x \bmod 2 = 1$. In the second step, it computes $\Pi(x)$=$(P_0 \cdot x + Q_x) \bmod N$, where $Q_x$ is equal to 1, 1+$N$/2+$P_1$, 1+$P_2$ or 1+$N$/2+$P_3$ when $x \bmod 4 = 0$, 1, 2 or 3 respectively. $P_0$, $P_1$, $P_2$ and $P_3$ are constants depending on $N$. Denote $\lambda(x) = P_0 \cdot x$, ARP interleaver can be efficiently implemented in a similar manner by reusing the same two ACC units, as shown in Fig. 6(b).

This architecture only requires a few adders and multiplexors which leads to very low complexity and can support all QPP/ARP turbo interleaving patterns. Compared to the latest work on row-column based interleaver [13] which needs complex state machines and RAMs/ROMs, our QPP/ARP interleaver architecture has lower gate count and more flexibility.

## 5. Parallel turbo decoder architecture

To increase the throughput, parallel MAP decoding can be employed by dividing the whole information block into several sub-blocks and then each sub-block is decoded separately by a dedicated MAP decoder [14][15][16][17]. For example, in [15] a 75.6 Mbps data rate is achieved using 7 SISO decoders running at 160 MHz.




(a) Normalized area versus throughput (N=6144, I=6, W=32)

(b) Normalized power versus throughput (N=6144, I=6, W=32)

**Figure 8. Architecture tradeoff for different parallelism and clock rates**

In this section, we propose a scalable decoder architecture, shown in Fig. 7(a), based on QPP/ARP contention-free 4G interleavers. The parallelism is achieved by dividing the whole block $N$ into $P$ sub-blocks (SBs) and assigning $P$ MAP decoders working in parallel to reduce the latency down to $O(N/P)$. The memory structure is designed to support concurrent access of LLRs by multiple ($P$) MAP decoders in both linear addressing and interleaved addressing modes. This is achieved by partitioning the memory into $Q$ banks ($Q \geq P$), with each bank being independently accessible. In this paper, we use $Q$=$P$. Take the QPP interleaver as an example, $P$ MAP decoders always access data simultaneously at a particular offset $x$, which guarantees no memory access conflicts due to the contention-free property of $\lfloor \Pi(x + jM)/M \rfloor \neq \lfloor \Pi(x + kM)/M \rfloor$, where $x$ is the offset in the sub-block $j$ and $k$ ($0 \leq j < k < P$), and $M$ is the sub-block length $N/P$. Since any MAP decoder will write to any memory during the decoder process, a full crossbar is designed for routing data among them. Fig. 7(b) depicts a parallel decoding example for duo-binary codes (general concepts hold for binary codes).

## 5.1. Architecture trade-off analysis

High throughput is achieved at a cost of multiple MAP decoders and multiple memory banks. In this section, we will analyze the impact of parallelism on throughput, area and power consumption. The maximum throughput is estimated as:

$$\text{Throughput} = \frac{N}{\text{Decoding time}} \approx \frac{N \cdot f}{2 \cdot I \cdot (\frac{\tilde{N}}{P} + 3\tilde{W})} \quad (8)$$

where $\tilde{N}$=$N$/2 and $\tilde{W}$=$W$/2 for Radix-4 decoding, $I$ is the number of iterations (contains two half iterations), $3\tilde{W}$ is the decoding latency for one MAP decoder and $f$ is the clock frequency. And the area is estimated as:

$$\text{Area} \approx P \cdot A_{\text{map}}(f) + A_{\text{mem}}(P, N) + A_{\text{route}}(P, f) \quad (9)$$

where $A_{\text{map}}$ is one MAP decoder area which increases with $f$, $A_{\text{mem}}$ is the total memory area which increases with $N$ and also $P$ due to the sub-banking overhead, and $A_{\text{route}}$ is the routing cost (crossbars plus interleavers) which increases with both $P$ and $f$. Note that the complexity of the full crossbar actually increases with $P^2$. We describe the decoder in Verilog and synthesize it on a 65 nm technology using Synopsys Design Compiler. The area tradeoff analysis is given in Fig. 8(a) which plots the normalized area versus throughput for different parallelism levels and clock rate (80-200MHz, step of 20MHz). The power estimation is obtained from the product of area and frequency (actual power estimation is still in progress). This estimation is based on the dynamic power consumption $P = CfV^2$ where $V$ is assumed to be the same for all the cases and $C$ is assumed to be proportional to the silicon area. Fig. 8(b) shows the power tradeoff analysis based on the $area \times f$ metric.

## 5.2. Comparison with related works

Table 3 compares the flexibility and performance with existing state-of-the-art turbo decoders. Fig. 8(a) compares the silicon area with [4][14][15][16] where the area is scaled and normalized to a 65nm technology, and the block size is scaled to 6144. In [2][4][18], programmable processors are proposed to provide multi-code flexibilities. In [14][15]

**Table 3. Comparison with existing turbo decoder architectures**

| Work | Architecture | Flexibility | MAP algorithm | Parallelism | Iteration | Frequency | Throughput |
|---|---|---|---|---|---|---|---|
| [2] | 32-wide SIMD | Multi-code | Max-LogMAP | 4 window | 5 | 400 MHz | 2.08 Mbps |
| [18] | Clustered VLIW | Multi-code | LogMAP/Max-LogMAP | Dual cluster | 5 | 80 MHz | 10 Mbps |
| [4] | ASIP-SIMD | Multi-code | Max-LogMAP | 16 ASIP | 6 | 335 MHz | 100 Mbps |
| [14] | ASIC | Single-code | LogMAP | 6 SISO | 6 | 166 MHz | 59.6 Mbps |
| [15] | ASIC | Single-code | LogMAP | 7 SISO | 6 | 160 MHz | 75.6 Mbps |
| [16] | ASIC | Single-code | Constant-LogMAP | 64 SISO | 6 | 256 MHz | 758 Mbps |
| This work | ASIC | Multi-code | LogMAP | 32 SISO | 6 | 200 MHz | 711 Mbps |

efficient ASIC architectures are presented for decoding of simple binary turbo codes using the LogMAP algorithm. In [16], a high speed Turbo decoder architecture based on the sub-optimal Constant-LogMAP SISO decoders (Radix-2) is discussed. Though the decoder in [16] achieves high throughput at a low silicon area, it has some limitations, e.g. the interleaver is non-standard compliant and it can not support WiMax duo-binary turbo codes. As can be seen, our architecture exhibits both flexibility and efficiency in supporting multiple turbo codes (simple binary + double binary) and achieving high throughput.

## 6. Conclusion

A flexible multi-code turbo decoder architecture is presented together with a low-complexity contention-free interleaver. The proposed architecture is capable of decoding simple and double binary turbo codes with a limited complexity overhead. A date rate of 10-711 Mbps is achievable with scalable parallelism. The architecture is capable of supporting multiple proposed 4G wireless standards.

## References


[1] C. Berrou, A. Glavieux, and P. Thitimajshima. Near Shannon limit error-correcting coding and decoding: Turbo-Codes. In *IEEE Int. Conf. Commun.*, pages 1064–1070, May 1993.

[2] Y. Lin et al. Design and implementation of turbo decoders for software defined radio. In *IEEE Workshop on Signal Processing Design and Implementation (SIPS)*, pages 22–27, Oct. 2006.

[3] M.C. Shin and I.C. Park. SIMD processor-based turbo decoder supporting multiple third-generation wireless standards. *IEEE Trans. on VLSI*, vol.15:pp.801–810, Jun. 2007.

[4] O. Muller, A. Baghdadi, and M. Jezequel. ASIP-Based multiprocessor SoC design for simple and double binary turbo decoding. In *DATE'06*, volume 1, pages 1–6, Mar. 2006.

[5] P. Robertson, E. Villebrun, and P. Hoeher. A comparison of optimal and sub-optimal MAP decoding algorithm operating in the log domain. In *IEEE Int. Conf. Commun.*, pages 1009–1013, 1995.

[6] M. Bickerstaff et al. A 24Mb/s radix-4 logMAP turbo decoder for 3GPP-HSDPA mobile wireless. In *IEEE Int. Solid-State Circuit Conf. (ISSCC)*, Feb. 2003.

[7] Y. Zhang and K.K. Parhi. High-throughput radix-4 logMAP turbo decoder architecture. In *Asilomar conf. on Signals, Syst. and Computers*, pages 1711–1715, Oct. 2006.

[8] C. Zhan et al. An efficient decoder scheme for double binary circular turbo codes. In *IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, pages 229–232, 2006.

[9] G. Masera, G. Piccinini, M. Roch, and M. Zamboni. VLSI architecture for turbo codes. In *IEEE Trans. VLSI Syst.*, volume 7, pages 369–3797, 1999.

[10] P. Salmela, R. Gu, S.S. Bhattacharyya, and J. Takala. Efficient parallel memory organization for turbo decoders. In *Proc. European Signal Processing Conf.*, pages 831–835, Sep. 2007.

[11] J. Sun and O.Y. Takeshita. Interleavers for turbo codes using permutation polynomials over integer rings. *IEEE Trans. Inform. Theory*, vol.51, Jan. 2005.

[12] C. Berrou et al. Designing good permutations for turbo codes: towards a single model. In *IEEE Conf. Commun.*, volume 1, pages 341–345, June 2004.

[13] Z. Wang and Q. Li. Very low-complexity hardware interleaver for turbo decoding. *IEEE Trans. Circuit and Syst.*, vol.54:pp. 636–640, Jul. 2007.

[14] M. J. Thul et al. A scalable system architecture for high-throughput turbo-decoders. *The Journal of VLSI Signal Processing*, pages 63–77, 2005.

[15] B. Bougard et al. A scalable 8.7-nJ/bit 75.6-Mb/s parallel concatenated convolutional (turbo-) codec. In *IEEE Int. Solid-State Circuit Conf. (ISSCC)*, Feb. 2003.

[16] G. Prescher, T. Gemmeke, and T.G. Noll. A parametrizable low-power high-throughput turbo-decoder. In *IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, volume 5, pages 25–28, Mar. 2005.

[17] S-J. Lee, N.R. Shanbhag, and A.C. Singer. Area-efficient high-throughput MAP decoder architectures. *IEEE Trans. VLSI Syst.*, vol.13:pp. 921–933, Aug. 2005.

[18] P. Ituero and M. Lopez-Vallejo. New schemes in clustered vliw processors applied to turbo decoding. In *Proc. Int. Conf. on Application-specific Syst., Architectures and Processors (ASAP)*, pages 291–296, Sep. 2006.